IJCTA, 9(27), 2016, pp. 165-173 ISSN: 0974-5572

Validation of Object Oriented Metrics Using Open Source Software System: Fuzzy Logic Technique

D.I. George Amalarethinam* and P.H. Maitheen Shahul Hameed**

ABSTRACT

Object-oriented technology has rapidly been accepted as the preferred paradigm for large-scale system design. This technology helps in developing software products of higher quality and lower maintenance cost. The Object-oriented metrics play an important role in the software development. Many metrics have been proposed related to various constructs like class, coupling, cohesion, inheritance, information hiding and polymorphism. But there is a little understanding of the empirical hypotheses and application of these measures. This research paper investigates Object-oriented metrics proposed by Chidamber and Kemerer (CK). These metrics are then applied to several java programs to analyze the complexity of a software product. This model is applied to predict the maintainability, cost and reusability factors. The MATLAB and Fuzzy logic approaches have been used for the assessment of complexity in Object-oriented Systems.

Keywords: Object-oriented Software Development, Software Metric, Fuzzy Inference System, Cohesion, and Coupling.

1. INTRODUCTION

Object-oriented programming (OOP) is a programming paradigm based on the concept of objects [1], which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures; often known as methods. Object-Oriented Analysis and Design (OOAD) of a software provide many benefits such as reusability, decomposition of problem into easily understood objects and the aiding of future modifications. But the OOAD software development life cycle is not easier than the typical procedural approach. Therefore, it is necessary to provide dependable guidelines that one may follow for ensuring good Object-oriented programming practices and write reliable code. Object-Oriented programming metrics is to be considered as an aspect. Metrics is a set of standards against which one can measure the effectiveness of Object-Oriented Analysis techniques in the design of a system [2]. Software complexity has been shown as one of the major contributing factors to the cost of developing and maintaining a software [17].

2. LITERATURE REVIEW

Several metrics have been proposed for Object-Oriented systems by researchers. A metric suite proposed by Chidamber and Kemerer (CK) [6] is one of the best known suites of Object-Oriented metrics. The six metrics proposed by CK are Weighted Method per Class (WMC), Depth of Inheritance Tree (DIT), Response For a Class (RFC), Number of Children (NOC), Lack of Cohesion of Methods (LOCM) and Coupling Between Objects (CBO) [1].

Morteza Asadi and Hassan Rashidi [3] proposed a new model to improve the maintainability of object-oriented software. This model is based on the newer versions of software quality standard and it is according to the measurement of several new metrics.

* Research Supervisor, Jamal Mohamed College, Tiruchirappalli.

** Research Scholar, Jamal Mohamed College, Tiruchirappalli.

Mansi et al. [4] focus on a set of Object-oriented metrics that can be used to measure the quality and effectiveness of an Object-oriented design. The metrics for Object-oriented design focus on measurements that are applied to the class and design characteristics. They also summarize the existing metrics, which will guide the designers to support their Object-oriented design.

Amjan Shaik, et al [7] proposed the current state of the art in Metrics and Object-Oriented Software System Quality and it gives the short descriptive taxonomy of the Object-Oriented Design and Metrics.

Arti Chhikara and R.S.Chhillar [8] investigate several Object-oriented metrics proposed by various researchers. These Object-oriented metrics are then applied to several Java programs to analyze the complexity of a software product.

3. CHIDAMBER AND KEMERER METRICS SUITES

CK et al. [6] proposed six metrics which have generated a significant amount of interest and are currently the most well known object-oriented suite of measurements for Object-Oriented software.

3.1. Weighted Methods per Class (WMC)

WMC measures the sum of complexity of the methods in a class. A predictor of the time and effort is required to develop and maintain a class that uses the number of methods and the complexity of each method. The high value of WMC indicates that the class is more complex as compared to the low values.

3.2. Depth of Inheritance Tree (DIT)

DIT metric is used to find the length of the maximum path from the root node to the end node of the tree. DIT represents the complexity and the behaviour of a class, and the complexity of design of a class and potential reuse.

3.3. Number of children (NOC)

According to Chidamber and Kemerer, the Number of Children (NOC) metric may be defined for the immediate sub class coordinated by the class in the form of class hierarchy [9,10].

3.4. Coupling Between Objects (CBO)

CBO is used to count the number of the classes to which the specific class is coupled. The rich coupling decreases the modularity of the class making it less attractive for reusing the class and the more high coupled class is more sensitive to change in other part of the design through which the maintenance is very difficult in the coupling of classes.

3.5. Response for class (RFC)

The RFC is defined as a set of methods that can be executed in response and messages received by the object of that class. Larger values also complicated the testing and debugging of the object through which, it requires the tester to have more knowledge of the functionality.

4. COMPLEXITY OF OBJECT-ORIENTED SYSTEM

Brooks [4] points out, that the complexity of software is an essential property and is not an accidental one. The Object-oriented decomposition process merely helps to control the inherent complexity of the problem. It does neither reduce nor eliminate the complexity. Measurement of the software complexity of Object-oriented systems has the potential to aid in the realization of these expected benefits.

Software complexity has been shown to be one of the major contributing factors to the cost of developing and maintaining software [6]. According to Coad and Yourdon [11], a good Object-oriented design is one that allows

trade-offs of analysis, design, implementation and maintenance costs throughout the lifetime of the system so that the total lifetime costs of the system are minimized. Software complexity measurement can contribute in minimising the cost trade-offs in two ways. They are:

1) To provide a quantitative method for predicting how difficult it will be to design, implement, and maintain the system.

2) To provide a basis for making the cost trade-offs necessary to reduce costs over the lifetime of the system.

5. EXPERIMENTAL CASE STUDIES

In order to validate the proposed model two medium/small size projects [16] are developed in Java. First project, the "Bank Project" has 10 components and Second project, the "CCMAT" (Cognitive Complexity Metric Analysis Tool) [11] has 7 components. The direct measurable metrics for CK Metrics Suite is automated by parsed Java code in Complexity Metric Analysis Tool (CMAT). Every time Java code is passed, the values of various metrics are calculated. The values of these metrics for all components are listed in Table 1 and Table 2 against the name of the component [13].

Table 1
Metrics value for internal characteristics of First Project in java

File_Name	*WMC*	*DIT*	*NOC*	*CBO*	*RFC*
Auth.java	2	1	1	2	0
Account.java	3	1	2	1	0
AccountSimpleImpl.java	3	2	3	1	0
AuthLogging.java	2	0	0	0	1
BankingAuth.java	4	1	2	0	0
InSufficientBalanceException.java	2	2	3	1	2
PreserveCheckedException.java	3	1	1	1	0
IndentedLogging.java	2	1	2	2	0
Test.java	4	2	2	0	1
BankingPermission.java	2	3	3	1	0

Table 2
Metrics value for internal characteristics of Second Project in java

File_Name	*WMC*	*DIT*	*NOC*	*CBO*	*RFC*
BackImage.java	2	1	1	1	0
First.java	3	0	0	0	1
HomePage.java	4	1	1	1	0
ImageText1.java	2	1	1	1	1
Metrics1.java	11	3	2	2	0
Metrics2.java	2	0	0	0	0
Metrics3.java	6	0	0	0	1

6. EVALUATION OF COMPLEXITY

In this section fuzzy inference system is used to find out the complexity of Object-oriented system [10]. WMC, DIT, NOC and RFC are selected to find the complexity of Object-oriented system [14].

All input values are categorized as low, medium and high. The output maintainability is categorized as Very Low, Low, Medium, High and Very High. A rule base is created by using all feasible arrangements of inputs.

Our fuzzy model 'Object-oriented Complexity' for assessment of Complexity records the impact of CK Metrics. Our model has been presented in Figure 1.

Complexity of Object-oriented system has been taken in the scale of 0 to 50 and member functions as Very Low, Low, Medium, High and Very High. All the five input parameters WMC, DIT, NOC, CBO, and RFC are taken on the scale of 0 to 30. The member functions of all the five input variables have been taken as Low, Medium, and High.

Figure 8 shows all possible combinations of input sets i.e. 3^5 (243) sets.

Rule viewer is shown in Fig 9. Considering the inputs as WMC = 15 (Medium), DIT = 15 (Medium), NOC = 15 (Medium), CBO = 15 (Medium) and RFC = 15 (Medium), along with fuzzy rules, the Complexity has been evaluated as 34.7, which is high (H).

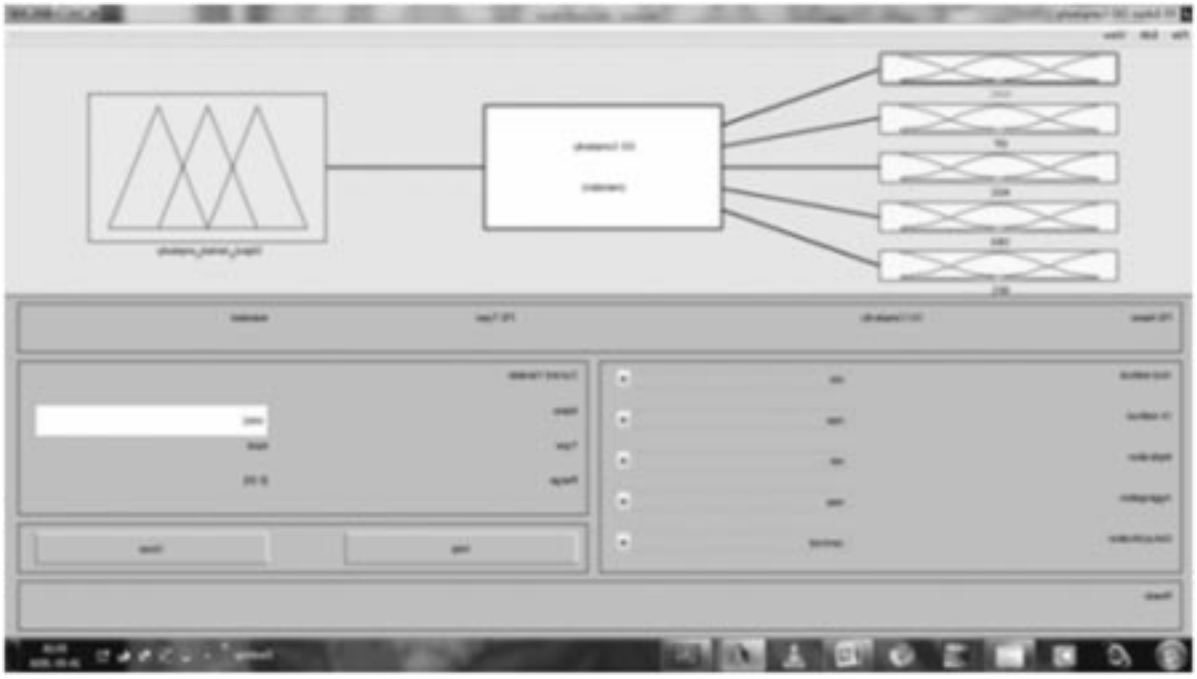

Figure 1: Fuzzy Model for assessment of Object-oriented Complexity

Figure 2: Membership functions for WMC

Figure 3: Membership functions for DIT

Figure 4: Membership functions for DIT

Figure 5: Membership functions for CBO

Figure 6: Membership functions for RFC

Figure 7: Membership functions for Object_Oriented_Complexity

Figure 8: Possible Combinations of Inputs (243 Rules)

Figure 9: Rule Viewer for Object Oriented Complexity

Table 3
Object-oriented Complexity for a given sets of input

Project Name	*WMC*	*DIT*	*NOC*	*CBO*	*RFC*	*Overall Complexity*
Bank Project	27[H]	14[M]	19[M]	9[L]	4[L]	25[M]
CCMAT	30[H]	6[L]	5[L]	5[L]	3[L]	17[L]

Figures 1 to 9 show the steps for fuzzy inference system to find the complexity of Object-oriented system.

The results for all set of possible input values (243 rules) are applied to the First Project and the Second Project. The results are tabulated in Table 3. For a set of inputs [33, 11, 14, 17, 4], the output is 25 (Medium) i.e. which means best complexity.

Complexity of Java applications can be evaluated using a variety of available software metrics from Software Engineering Domain. This research has presented a set of CK metrics [6] that can be used to project their complexity values, to assess testability and maintainability of the components. This evaluation concludes that there should be a compromise among internal software attributes in order to maintain a high degree of reusability while keeping the degree of complexity and coupling as low as possible [14].

7. STATISTICAL ANALYSIS

Table 4 shows the statistical values calculated for the metric values obtained from all components in the First Project and the Second Project.

Table 4
Statistical Values Calculated for all Components in Project

Metric Type	*Min*	*Max*	*Mean*	*Median*	*Stand. Deviation*
WMC	2	11	3.35294	3	2.262222
DIT	0	3	1.17647	1	0.951006
NOC	0	3	1.41177	1	1.064121
CBO	0	2	0.82353	1	0.727607
RFC	0	2	0.41176	0	0.618347

7.1. Observations

- Weighted Method per Class metric predicts time and effort that are required to build and maintain a class. A high value of WMC has been found to lead to more faults.
- In DIT, Deeper trees constitute greater design complexity, since more methods and classes are involved, but at the same time reusability also get increased due to inheritance.
- The greater the number of children, the greater the likelihood of improper abstraction of the parent and may be a case of misuse of sub classing. However, high NOC indicates high reuse, since inheritance is a form of reuse.
- RFC metric looks at the combination of the complexity of a class through the number of methods and the amount of communication with other classes. From this study, it was found that components are less complex as the mean value of this metric is low for components.

8. CONCLUSION AND FUTURE WORK

This research paper proposed a set of CK metrics that can be used to rank programs on their complexity values, to assess testability and maintainability of the programs. This paper concludes that there should be a compromise among internal software attributes in order to maintain a high degree of reusability while keeping the degree of complexity and coupling as low as possible. However, it is still insufficient and it needs further in-depth study. This work can be focused on empirical validation of Object-oriented metrics in multi languages environment in future. This analysis can be used as a reference by software developers for building a fault free, reliable, and easy to maintain software product in Java.

The study is conducted using open source tools and is applied on open source software system. So it will help in supporting the industrial acceptability of software metrics.

REFERENCES

[1] Dr. D.I. George Amalarethinam and P.H. Maitheen Shahul Hameed, "Analysis of Object- oriented Metrics on a Java Application", International Journal of Computer Applications (IJCA), **1**, 0975-8887, 2015.

[2] Dr. D.I. George Amalarethinam and P.H. Maitheen Shahul Hameed, "Factors Influencing Software Quality", in Reflection – des ERA Journal of Mathematical Science, 2012.

[3] Morteza Asadi and Hassan Rashidi, "A Model for Object-Oriented Software Maintain-ability Measurement" *I.J. Intelligent Systems and Applications*,**1**, 60-66, 2016.

[4] Mrs. Mansi Aggarwal , Dr. Vinit Kumar Verma, Mr. Harsh Vardhan Mishra, "An Analytical Study of Object-Oriented Metrics", International Journal of Engineering Trends and Technology (IJETT) – **6(2)**, 2013.

[5] Harsha Singhani, Dr. Pushpa R. Suri, "Object-oriented Softwar Testability (OOSTe) Metrics Assessment Framework", IJARCSSE, **4**, 1096-1106, 2015.

[6] S.R.Chidamber and C.F. Kemerer. "A metrics suite for Object-oriented design", IEEE transactions on Software Engineering, **20(6)**, 476-493, 1994.

[7] Lov Kumar, Debendra Kumar Naik, Santanu Ku. Rath, "Validating the Effectiveness of Object-Oriented Metrics for Predicting Maintainability", ICRTC, Elsevier, 2015.

[8] Arti Chhikara and R.S.Chhillar, "Analyzing the Complexity of Java Programs using Object-oriented Software Metrics ", IJCSI International Journal of Computer Science Issues, **9(1)3**, 2012.

[9] Kalpana Johari, Arvinder Kaur, "Validation of Object-oriented Metrics Using Open Source Software System: An Empirical Study", ACM SIGSOFT, **37(1)**, 2012.

[10] P.K. Singh, O.P.Sangwan, A.P. Singh, A. Pratap, "An Assessment of Software Testability using Fuzzy Logic Technique for Aspect-Oriented Software", IJITCS, **3**, 18-26, 2015.

[11] Kulwinder Singh, P.K. Bhatia, "An Approach to Measure Cognitive Complexity for Object-oriented code", IJRASET, **3(1)**, 2015.

[12] Dr. Herbert Raj, Dr. A. Aloysius, Dr.L.Arockiam, "Cognitive Complexity Metrics Analysis Tool (CCMAT)", IJETCCT, **1(1)**, 2014.

[13] Ritika Chaudhary, Ram Chatterjee, "Reusability in AOSD – The Aptness,Assesment and Analysis", ICROIT, 2014.

[14] Aloysius, "A cognitive complexity metrics suite for Object-oriented design" Thesis, 2012

[15] Mansi Aggarwal, Dr. Vinit Kumar Verma, Harsh Vardhan Mishra, "An Analytical Study of Object-Oriented Metrics" IJETT, **6(2)**, 2013.

[16] Open source software, www.sourceforge.net

[17] Roger S.Pressman:Software Engineering, A practioner's Approach, Fifth Edition, 2001.

[18] V. R. Basili, L. Briand, and W. L. Melo, "A validation of object-oriented design metrics as quality indicators," IEEE Transactions on Software Engineering, **22**, pp. 751–761, 1996.

[19] Misra S., "An Approach for Empirical Validation Process of Software Complexity Measures", In press, Acta Polytechnica, Hungarica. **4**, 2011.